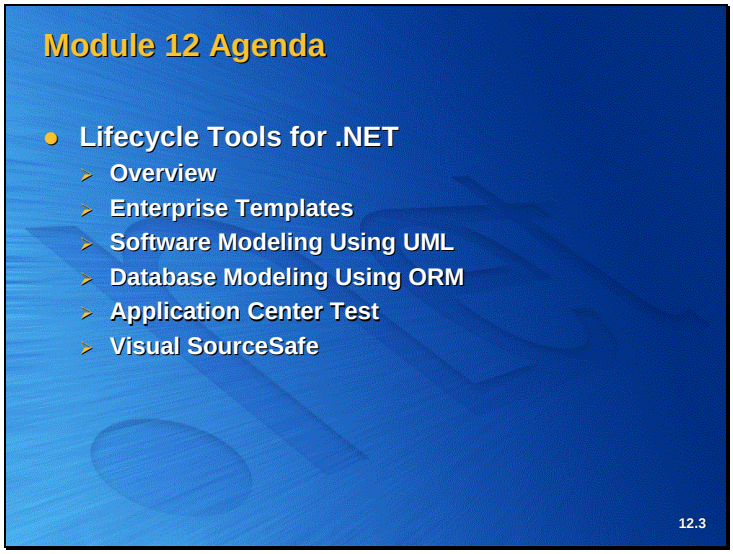
Microsoft
.net developer
training
Create XML Web Services and Applications with Visual Studio .NET

Lifecycle Tools for .NET

Agenda: Day 2

- Module 09: Creating XML Web Services
- Module 10: Code Instrumentation and Middle Tier Technology
- Module 11: Security in .NET
- ▶ Module 12: Lifecycle Tools for .NET
- Module 13: Configuring and Deploying .NET Applications
- Module 14: Application Interoperability
- Module 15: Migrating Applications to .NET
- Module 16: Building Applications for Mobile Devices

 ©2002 Microsoft Corporation



Module 12 Agenda

What you will learn:

- Enterprise Templates
- Software Modeling using UML
- Database Modeling using ORM
- Application Center Test
- Visual SourceSafe (VSS)

Lifecycle Tools for .NET – Overview

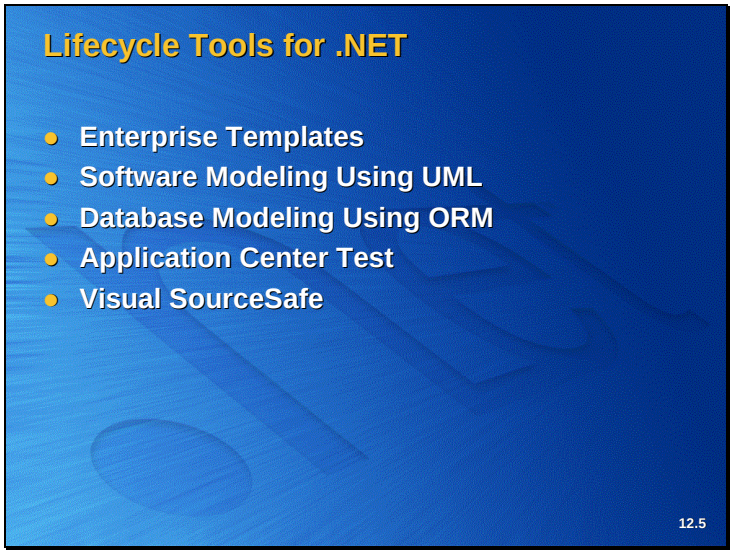
Enterprise product development, in many ways, is very similar to traditional development. All development work requires state of the art compilers and designers and those types of productivity tools. But enterprise customers tend to be very systematic as opposed to opportunistic in their approach to building applications.

Successfully building complex enterprise applications requires a solid architecture and a common understanding of requirements across the development team. Enterprise development organizations typically require a collaborative work environment where there is a need for different team members to work closely, share and communicate information.

Enterprise development teams typically have two key roles in their development teams: the developer and the architect. While developers are responsible for the implementation of the project, the architects (who may be senior/expert developers) are responsible for helping their organizations build applications that scale, integrate with existing systems, fulfill the business requirements, and be maintainable over multiple versions of the product. So, in order to help customers build robust enterprise applications from the ground up, we think it is important to provide tools that help build solid architectures.

Additionally, having standards for architectural design and development is important. Reusable assets in the way of templates and architectural guidance are essential to building enterprise-level products in a systematic, repeatable and predictable manner.

 ©2002 Microsoft Corporation



Lifecycle Tools for .NET

Visual Studio .NET for the Enterprise comprises of:

- Enterprise Architect
- Enterprise Developer

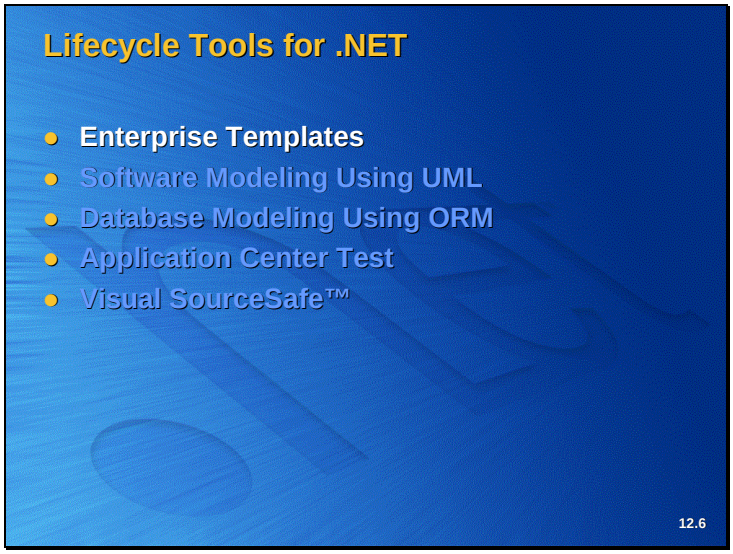
These products provide exciting and innovative life-cycle tools for designing, building and testing XML Web services and applications in a collaborative team environment.

The lifecycle tools for .NET include:

- Enterprise Templates
- Software Modeling using UML
- Data Modeling using ORM
- Application Center Test
- Visual SourceSafe

Built-in Flexibility of Visio

In Visual Studio .NET, the Database and Software modeling tools are based on Visio®. Tight integration of Visio with Microsoft Office allows designs to be shared in Word documents, PowerPoint presentations, or in Web pages. Models are not restricted to a fixed set of symbols and can be annotated with a variety of shapes.



Lifecycle Tools for .NET

- ▶ **Enterprise Templates**
- Software Modeling Using UML
- Data Modeling Using ORM
- Application Center Test
- Visual SourceSafe

 ©2002 Microsoft Corporation

Enterprise Templates
Goals
Share corporate knowledge & experience at design time to:
Reduce the "Blank Slate" problem
Reduce the "Full Slate" problem
Make it easier to do the right thing
12.7

Enterprise Templates – Goals

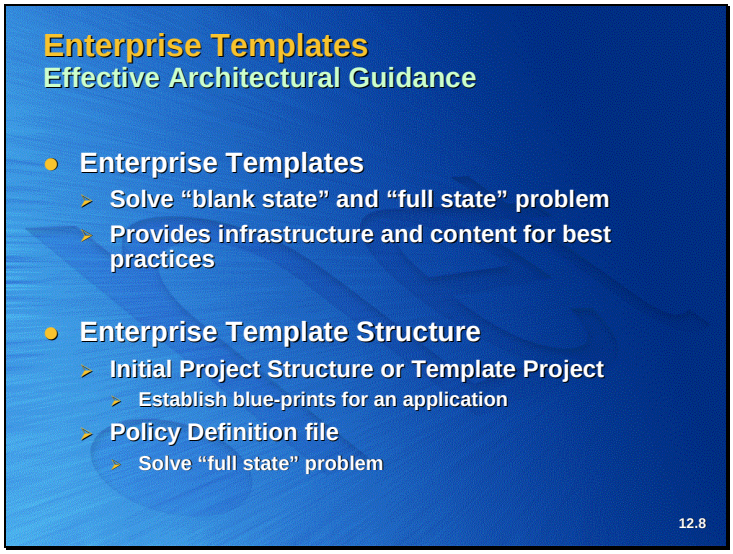
Many organizations have an "architecture" or "best practices" group to learn the technology, and figure out the best way to apply it within their organization. The output of this is a document, or a Web site with reams of information. How realistic is it that all the developers are going to wade through this material, and digest it, before they ever start programming? It never happens.

So what do you do? If best practices aren't widely adopted, they provide no value. Adopting best practices affects how efficiently and cost effectively the business can be managed. Coming up with best practices is actually lost productivity if no one follows those practices. Those developers should have just written the code rather than coming up with best practices.

For Example, **File | New | Project**, starts you off with a blank form. But then what? How do you develop applications with the same look and feel as other applications in your organization? How do you ensure consistency? If you're new on the job, how do you even get started?

In addition, you have hundreds of options in your project workspace. These include the toolbox, with all it's icons, hundreds of menu items and hundreds of properties. Maybe you know what a few of these are. Maybe you are consistently using some of them. Everything else is just noise. We call this the 'blank slate' problem or the '747 Cockpit' problem.

If you're building a business rule component, we want to give you a way to say "for business rules components, these are the things that make sense", and have these rules applied even to people who are new on the job. For example, it doesn't make sense to add a user interface component to the business rule. Sometimes, there are certain controls that you always configure in a certain way. In other words, you have your own defaults that are different from what the inventors of visual studio had in mind.



Effective Architectural Guidance

Enterprise Templates provide a powerful way to solve both the "blank slate" (no starting point) and the "full slate" (too many alternatives) problems. This innovative set of features enables organizations to provide architectural guidance within the Visual Studio .NET environment.

Enterprise templates enable companies to leverage their most experienced people to define development guidance and policy that can be easily used by developers for building applications. Instead of being limited to documents, this guidance can take the form of architectural blueprints, reusable components and policies or instructions for building applications — all delivered within the Visual Studio .NET environment.

Enterprise Templates
The Visual Studio .NET enterprise templates provide key infrastructure and content to define and deliver best practices and development policies in an organization.

Enterprise Template Structure

Enterprise Template encompasses two parts:

1. **An initial project structure or the template project:**

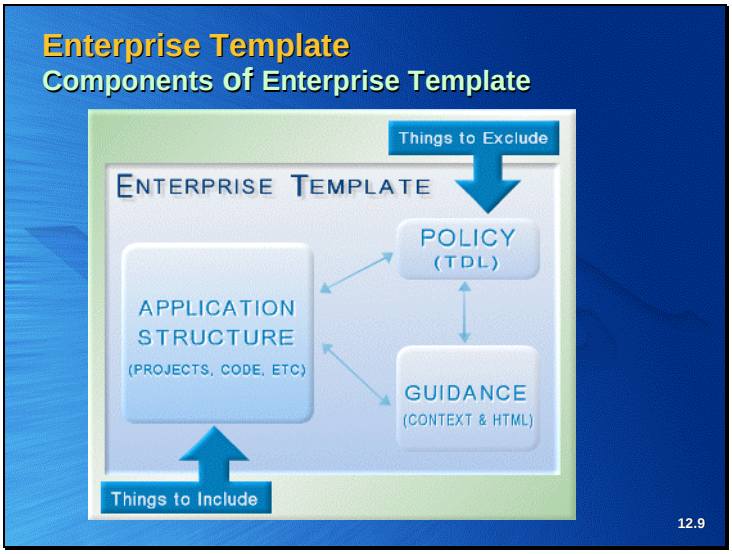
   Using template projects, architects can establish the blueprints for an application. In addition to specifying this initial structure for an application, an application architect can include in the Template Projects' standard components and any other reusable assets that are appropriate for the application.

   Visual Studio .NET includes the Simple Distributed Application and the more complex Distributed Application predefined Template Projects. Both are available with Visual Basic .NET or Visual C# .NET as the default component language. These projects can be used as is or customized and saved as new project types that are available within the New Project dialog box.

2. **A policy definition file:**

   Policy is provided using a new XML-based language, Template Description Language (TDL).

 ©2002 Microsoft Corporation



Components of Enterprise Template

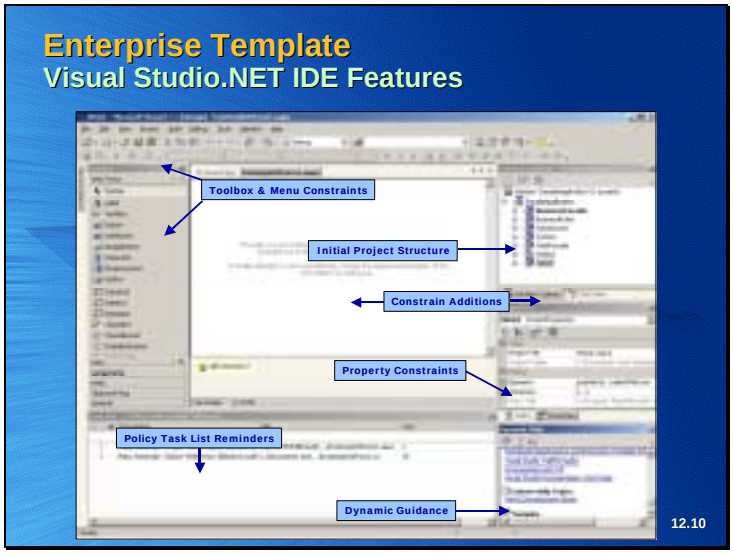
Enterprise Templates comprise of the components: Application Structure, Policy and Guidance.

Application Structure
The initial set of things that you want to include, or make available, to someone building a solution. It contains items that create the high-level structure that can be used as a starting point such as projects, components, and classes. This is most powerful if you have a reusable framework to guide people to use.

Policy
The policy describes things that should be excluded. The policy file is defined in TDL, which is used to specify the technologies that are appropriate at various points in a project. The policy component associated with a template project provides the solution to the full-slate problem. TDL also provides fine-grained control of property settings. TDL enables an architect to control the Visual Studio .NET environment, including the task list, toolbox, property browser, solution explorer, designers and editors, and even dynamic help
When a developer uses the template project, its associated policy is automatically applied within the context of the Visual Studio .NET environment. Policy can even be attached to pre-existing projects, with any violations referenced in the task list for easy correction.

Guidance
You can provide custom help that integrates with Visual Studio, to provide the right help at the right time.

Using the prototype defined, the template services, which is a set of DLLs, creates the initial structure, and then the policy is applied. The policy interacts with the various features in the Visual Studio IDE (such as the TaskList, Toolbox, Property Browser, etc.) to customize for the developer and configures the development environment to guide people in the right direction.



Visual Studio.NET IDE Features

Visual Studio.NET IDE helps to configure the development environment to guide people in the right direction and to develop the right kind of applications for the organization.

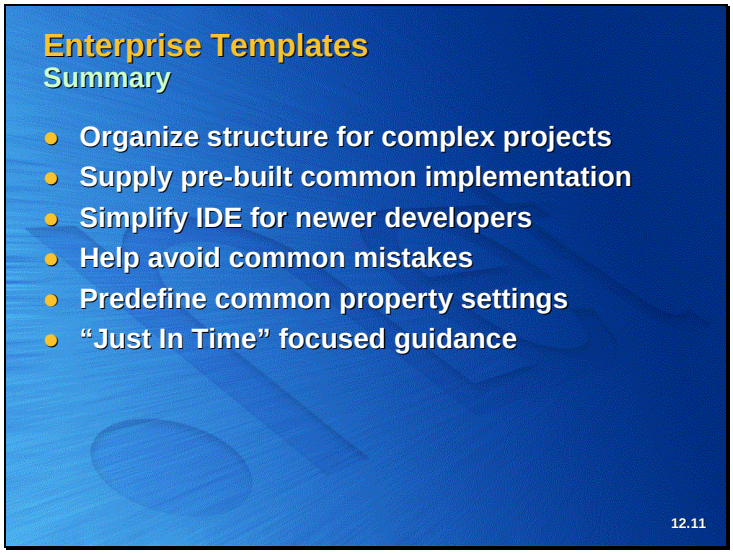
We can define the initial project structure to include projects, DLLs, XML schemas, and implementation code. It could choose to structure the initial project such that components separate business logic from application layer and data layer. Using the prototype defined, the Template service, which is a set of DLLs creates the initial structure.

Depending on how the structure is defined and number of components, the context sensitive help can be set in HTML files and make them part of the dynamic help that is available from within Visual Studio. NET. When the developer selects a component in the project structure, help topics on that component and its usage are displayed in the Dynamic help window.

Once the project is defined, policies and constraints are applied on the components of the project so that they can be used only in certain ways. For example, if we want to avoid references from data layer to the web components, we can set up policies to disallow these types of references. If we make a reference that has been disallowed by the policy, policy reminders show up in the task list.

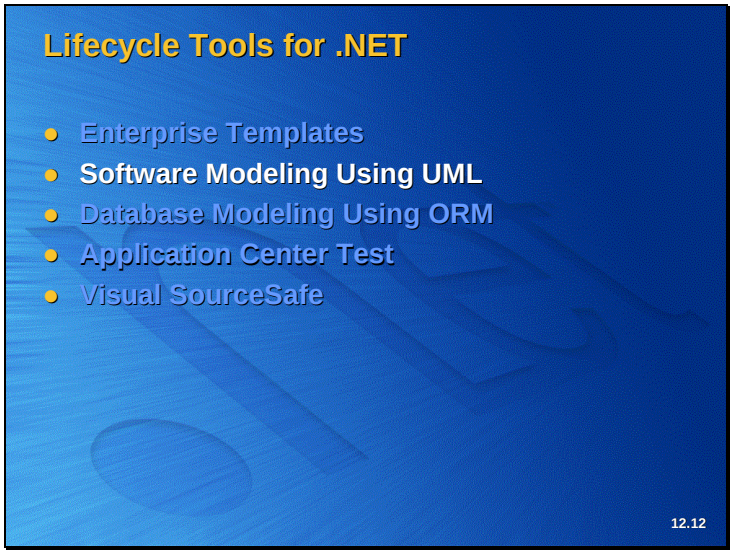
In addition to constraints in the project structure, we can customize the Visual Studio.NET IDE to show only items that are required for the project. We can set constraints in the property browser to default to Flow Layout, or customize the appearance of a DataGrid. Also, we can control the policies and restrict the values of certain properties to certain ranges.

 ©2002 Microsoft Corporation



Enterprise Templates – Summary

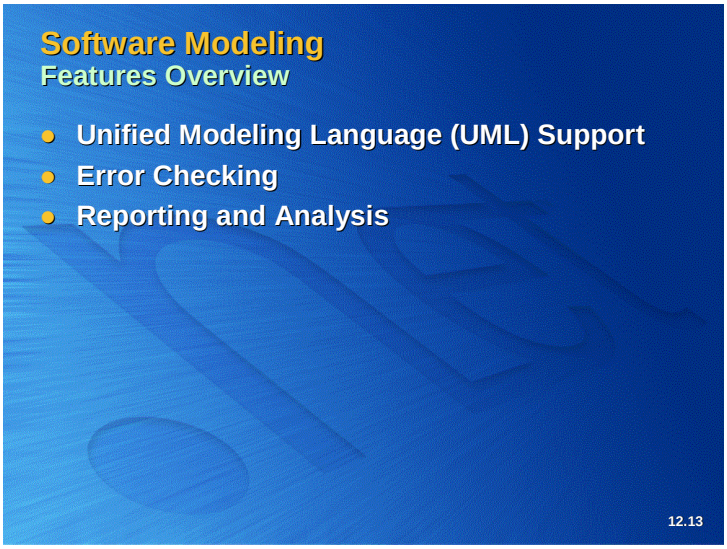
- Provides help to capture "best practices", and make them easily available to the developers.
- Provides a common structure and organization for applications.
- Provides the developer with some pre-built implementation, so developers build on top of the same base classes.
- Simplifies the IDE for new developers, and eliminate choices for certain contexts, where the organization has decided that those choices are not required.
- Provides pre-defined common property settings.
- Provides context sensitive help so that it provides guidance on time.



Lifecycle Tools for .NET

- Enterprise Templates
- ▶ **Software Modeling using UML**
- Data Modeling using ORM
- Application Center Test
- Visual SourceSafe

 ©2002 Microsoft Corporation



Software Modeling – Features Overview

Visual Studio .NET Enterprise Edition delivers support for a full range of design and modeling activities, including the free-form diagramming capabilities of Visio as well as a set of industry-standard Unified Modeling Language (UML) diagrams.

With the Visual Studio .NET modeling features, users can create sophisticated diagrams to specify their application architecture and business requirements as well as communicate these across their teams. Business analysts, architects, developers and others who want to perform analysis and design tasks to enhance communication and increase the productivity of their development teams can take advantage of these new capabilities.

Software Modeling features in Visual Studio .NET:

Unified Modeling Language (UML) Support
UML is a notation for visually describing and interpreting the pieces, relationships and actions that make up a software application.
Full support for the industry-standard UML 1.2 diagram notation provides a common meta-model for models created in Visual Studio .NET, so that changes made in one diagram can be reflected in all others.

Error Checking
Visio built-in error checking capabilities will catch both programmatic and semantic errors. Programmatic errors include attributes or operations without name properties, or defining incompatible data types for a generated language. For instance, Visio would generate a code error if an attribute with a data type of C++ unsigned integer exists in a class to be generated in VB.NET. Visio attempts to convert the unsigned integer, but cannot find a compatible VB.NET type.

Semantic errors include syntax errors such as missing data or improper use of the UML notation. Duplicate class names and relationships that do not connect to any other objects are examples of semantic errors.

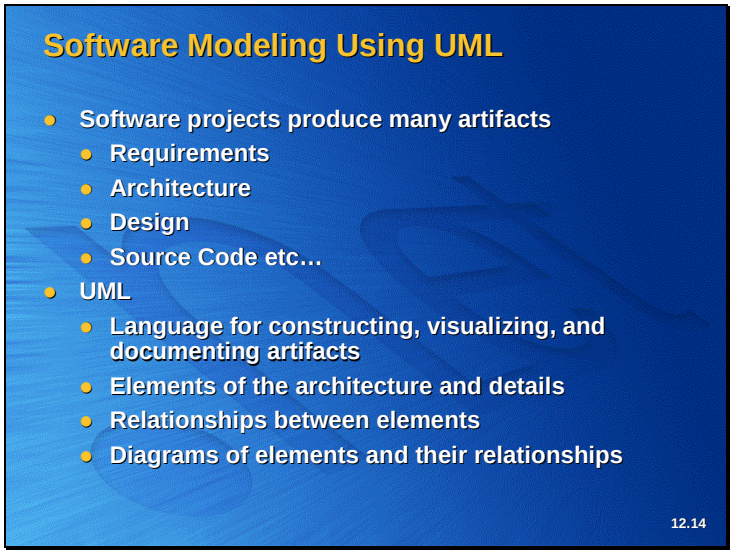
Using Visio, error checking of UML diagram can be done automatically or manually through a menu command. Code checking can be invoked manually or automatically while generating code from a UML diagram. Semantic error checking can be set up to run automatically, invoked manually through a menu command, or done automatically when diagram is exported.

Visio highlights the offending object in red, and lists all error messages in the Output window. From the Output window, you can quickly locate the source of the errors by double clicking on each message.

Reporting and Analysis
Visual Studio .NET Enterprise provides a rich set of standard reports thus making analyzing models efficient and straightforward.

Visio allows a great degree of control over the contents of UML reports. The scope of the report can be limited to include only the items within the active diagram or can report on all of the objects within the UML model. For each type of report, there are certain subcategories of items that you may include in the output. For an even deeper level of detail, you can indicate for each category, which properties should be included as well. The report can be made more readable by adding or removing separators between new logical groups of data.

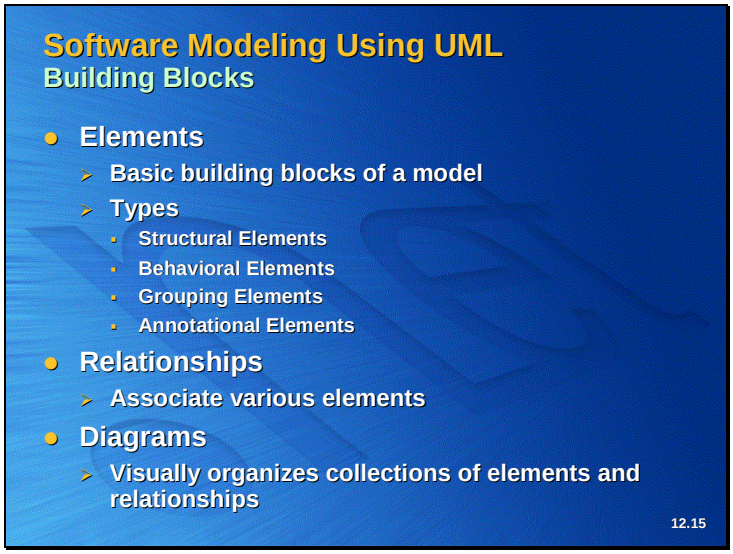
 ©2002 Microsoft Corporation



Software Modeling using UML

Software projects tend to produce a many artifacts during the design, implementation, and review process of software development. The Requirements documents, Design Documents, Project plans, Prototypes are the integral part of a software development project. UML gives expression to all of these through standardized constructs that can be implemented in Visual Studio. NET Enterprise.

UML is a language for visualizing, specifying, constructing, documenting the artifacts of a software intensive system.



Software Modeling using UML – Building Blocks

UML diagrams consist of three fundamental building blocks: Elements, Relationships and Diagrams that are discussed below.

Elements

Elements are the basic building blocks of a model. For example, a class would be a basic element.

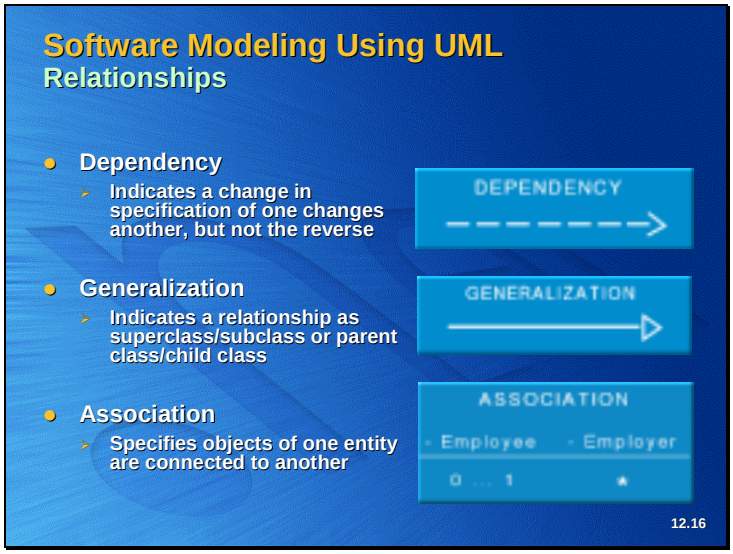
There are four types of elements present in UML. They are:

1. Structural elements – Includes class, component, use case, node, interface
2. Behavioral elements – Includes state and message.
3. Grouping elements – Includes packages
4. Annotational elements: Includes notes

Relationships
The relationship objects tie the various elements together in the diagram.

Diagrams:
Diagrams visually organize collection of elements and relationships.

 ©2002 Microsoft Corporation



Relationship elements in a UML Model

Relationships specify how different parts of the model interact. The types of relationships that you can create in UML are:

- Dependency
- Generalization
- Association

Dependency relationship
Dependency relationships specify that a change in a parent object might affect a change in a child object but not the opposite.

Generalization relationship
Generalization relationships specify that a sub-class is a specialization of a parent class. For example, a base stream class might form the foundation for numerous other classes such as IOStream, NetworkStream, MemoryStream etc as found in the System.IO namespace in the .NET framework. Another example of generalization would be a base shipper class that contains methods, properties, events, etc which other shipper classes (CareAir, AllAir, etc.) inherit from.

Association relationship
Association relationships specify that you can use the association name to describe the nature of the relationship. You can give the association name a direction to avoid ambiguity. When a class participates in an association, it has a specific role that it plays in the relationship. You can name the roles of an association.

An association represents a structural relationship among object. It can be important to state, how many objects may be connected across the association.

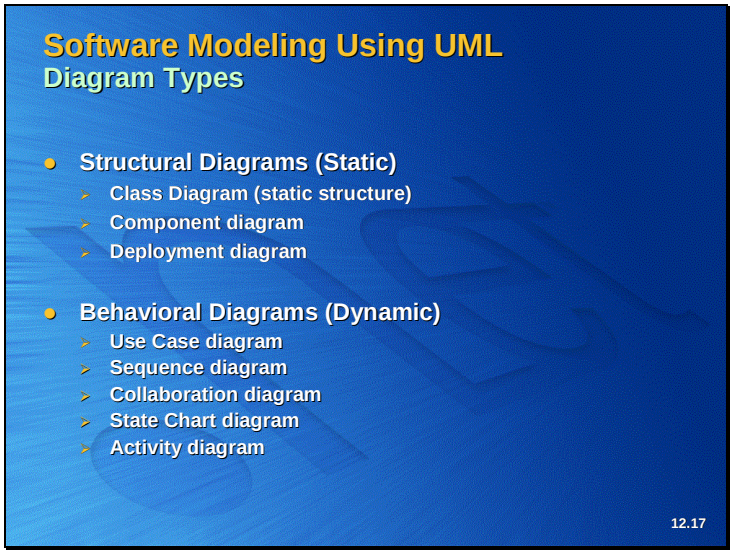


Diagram types in UML

The different types of UML Diagrams are:

1. Structural Diagrams
2. Behavioral Diagrams

Structural Diagrams

The Structural diagrams are used to visualize, specify, construct and document the static aspects of the system. The structural diagrams are to a software system what walls, doors, windows, and pipes are to a house. The Structural Diagrams include:

Class Diagrams

A class diagram shows a set of classes, interfaces, and collaborations and their relationships. Class diagrams are the most common diagrams found in modeling object-oriented systems. Class diagrams are used to illustrate the static design view of a system.

Component Diagrams

A component diagram shows a set of components and their relationships. Component diagrams are used to illustrate the static implementation view of a system. Component diagrams are related to class diagrams in that a component typically maps to one or more classes, interfaces, or collaborations.

Deployment Diagrams

A Deployment diagram shows a set of nodes and their relationships. Deployment diagrams are used to illustrate the static deployment view of an architecture. Deployment diagrams are related to component diagrams in that a node typically encloses one or more components.

Behavioral Diagrams

The five Behavioral diagrams are used to visualize, specify, construct and document the dynamic aspects of a system. Just as the dynamic aspects of a house encompass airflow and traffic through the rooms of a house, so too do the dynamic aspects of a software system encompass such things as the flow of messages over time and the physical movement of components across a network.

 ©2002 Microsoft Corporation

The five Behavioral diagrams include:

Use Case Diagrams

A Use Case diagram shows a set of use cases and actors (a special kind of class) and their relationships. Use Case diagrams are used to illustrate the static use case view of a system. Use case diagrams are especially important in organizing and modeling the behaviors of a system.

Sequence Diagrams

A Sequence diagram is an interaction diagrams that emphasizes the time ordering of messages. A sequence diagram shows a set of objects and the messages sent and received by those objects. The objects are typically named or anonymous instances of classes, but may also represent instances of other things, such as collaborations, components, and nodes. Sequence diagrams are used to illustrate the dynamic view of a system.

Collaboration Diagrams

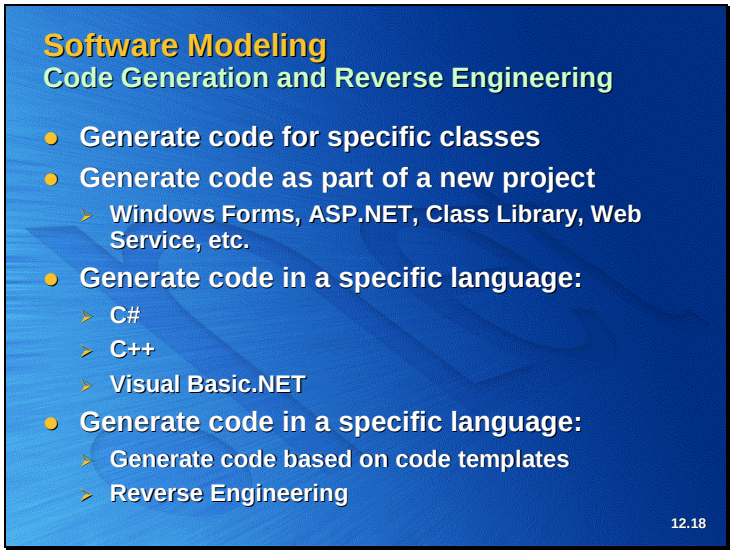
A Collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. A collaboration diagram shows a set of objects, links among those objects, and messages sent and received by those objects. The objects are typically names or anonymous instances of classes, but may represent instances of other things, such as collaborations, components, and nodes. Collaboration diagrams are used to illustrate the dynamic view of a system.

State Chart Diagrams

A State Chart diagram is a representation of a “state machine”. It is the representation of the states, events, and transitions that an object goes through in its lifetime.

Activity Diagrams

An Activity diagram shows the flow from activity to activity within a system. An activity shows a set of activities, the sequential or branching flow from activity to activity, and objects that act and are acted upon. Activity diagrams are used to illustrate the dynamic view of a system. Activity diagrams are especially important in modeling the function of a system. Activity diagrams emphasize the flow of control among objects.


Software Modeling
Code Generation and Reverse Engineering
Generate code for specific classes
Generate code as part of a new project
Windows Forms, ASP.NET, Class Library, Web Service, etc.
Generate code in a specific language:
C#
C++
Visual Basic.NET
Generate code in a specific language:
Generate code based on code templates
Reverse Engineering
12.18

Code Generation and Reverse Engineering

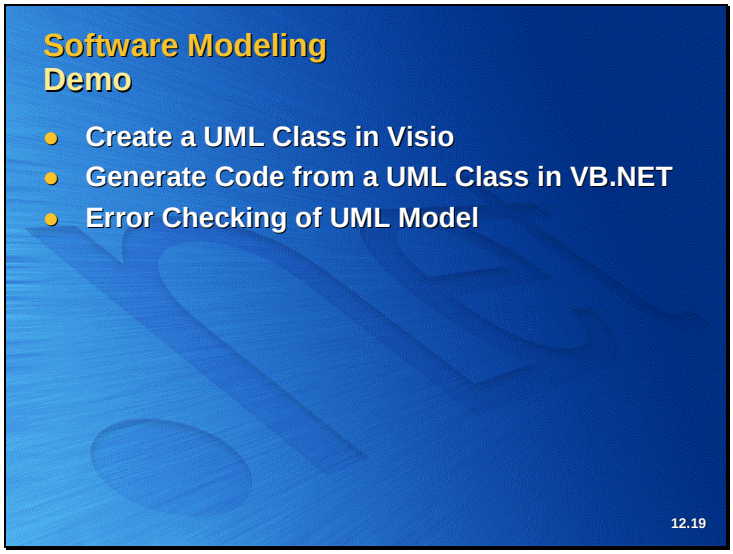
Visual Studio .NET Enterprise provides a very powerful and flexible way to turn UML diagrams into code skeletons.

Generated code can be based on one or more classes from a model and can be created as part of an existing project or as a stand-alone file. Also, it provides the ability to generate code in a specific language - C#, C++ or VB.NET.
Code templates can be used to control the format of the generated output and maintain consistency.

The ability to reverse engineer a class definition into UML models is built-in directly into the Visual Studio.NET Enterprise Edition. This model will contain complete class definitions imported from the specified project code.

The reverse engineering feature is available for C#, C++ and VB.NET. Any class variable, attributes, methods and even references for the classes will be imported to the Visio model.

 ©2002 Microsoft Corporation



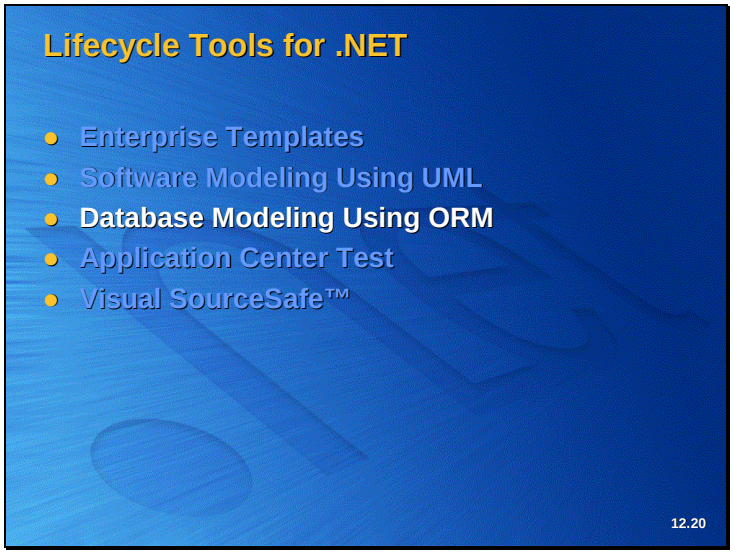
Software Modeling

Demo

Generating code from a UML class using Visio and Error Checking

Objectives:

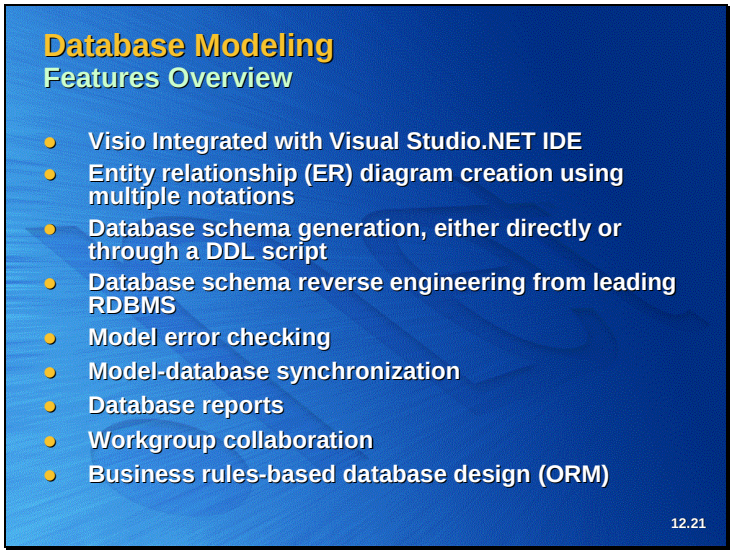
- Create a UML Class in Visio
- Generate Code from a UML Class in VB.NET
- Error Checking of UML Model



Lifecycle Tools for .NET

- Enterprise Templates
- Software Modeling using UML
- ▶ **Database Modeling using ORM**
- Application Center Test
- Visual SourceSafe

 ©2002 Microsoft Corporation


Database Modeling
Features Overview
Visio Integrated with Visual Studio.NET IDE
Entity relationship (ER) diagram creation using multiple notations
Database schema generation, either directly or through a DDL script
Database schema reverse engineering from leading RDBMS
Model error checking
Model-database synchronization
Database reports
Workgroup collaboration
Business rules-based database design (ORM)
12.21

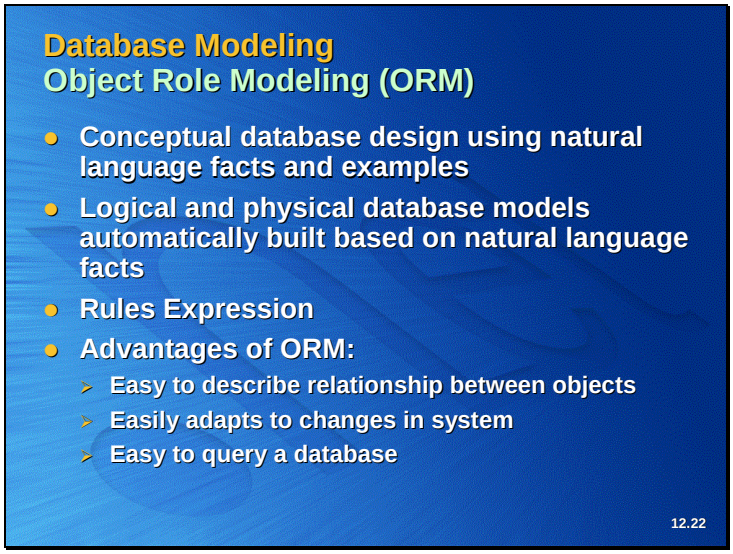
Database Modeling – Features Overview

Visual Studio.NET Enterprise Edition provides a set of database modeling tools, including conceptual modeling using Object Role Modeling, enabling business rules to be captured using simple English statements and then flowed into the database design process. It also supports industry standard ER and IDEF1X logical models, and generates physical schema for a number of relational databases.

For complex applications, the need to plan efficiently becomes more important. Modeling lets designers and developers work out the physical, logical, and conceptual workings of an application, and generate a skeleton for a developer to fill in the actual code. As solution to this, Visio is now fully integrated within the Visual Studio.NET Enterprise.

Visio allows generation of a physical database from a model or through DDL. Also, it supports detailed error checking and synchronization between the database model and the physical database. There is also a good amount of reporting and workgroup collaboration provided through built-in reports and the ability to merge different Visio files into one.

With Visio fully integrated within the Visual Studio.NET IDE, code can be reverse engineered inside Visual Studio .NET. In addition, diagrams are automatically added to the solution and hence easily accessible inside the Visual Studio.NET IDE.


Database Modeling
Object Role Modeling (ORM)
Conceptual database design using natural language facts and examples
Logical and physical database models automatically built based on natural language facts
Rules Expression
Advantages of ORM:
Easy to describe relationship between objects
Easily adapts to changes in system
Easy to query a database
12.22

Object Role Modeling (ORM)

Object Role Modeling provides a conceptual, easy-to-understand method of modeling data. The ORM methodology is based on three core principles:

- Simplicity: Data is modeled in the most elementary form possible.
- Communicability: Database structures are documented using language that can be easily understood by everyone.
- Accuracy: A correctly normalized schema is created based on the data model.

ORM has the ability to generate logical database model from the facts that you derive from the business rules. ORM database model allows you to generate logical and physical database models based on the natural language facts. The main idea behind ORM is to make the model more intelligible to the non-technical users.

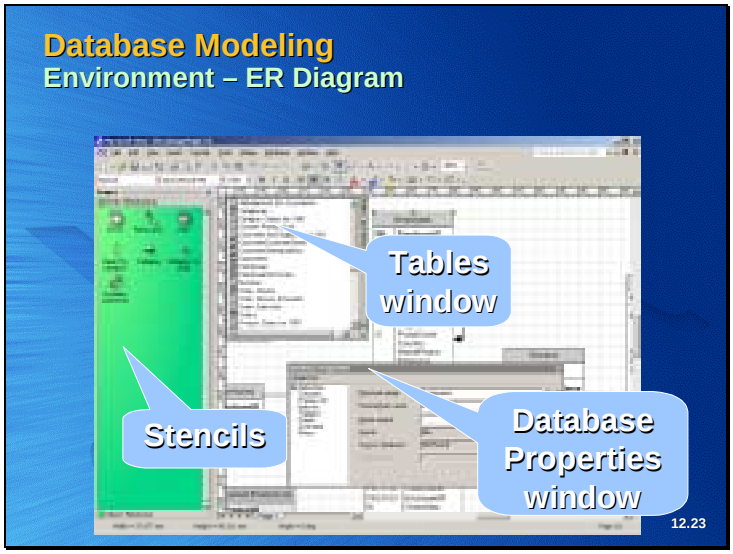
Rules Expression

ORM represents the application world as a set of objects (entities or values) that play roles (parts in relationships). ORM is sometimes called fact-based modeling because it verbalizes the relevant data as elementary facts. These facts can't be split into smaller facts without losing information.

Advantages of ORM

1. ORM provides a simple, direct way of describing relationships between different objects.
2. Models created using ORM have a greater capacity to adapt to changes in the system than those created using other methodologies. ORM allows reuse of objects and the data model automatically maps to a correctly normalized database schema.
3. The simplicity of the ORM model eases the database querying process. Using an ORM query tool, users can access the desired data without having to understand the underlying structure of the database.

 ©2002 Microsoft Corporation



Database Modeling – Environment for ER Diagram

Entity relationship (ER) is another type of database modeling. ER diagrams are an important part of the modeling process as they allow you to better visualize key database elements such as indexes, keys, and triggers in the model.

ER diagrams can be generated from the ORM diagrams. After adding appropriate elements to the ER diagram a physical database can then be easily generated from the embedded information in the diagram.

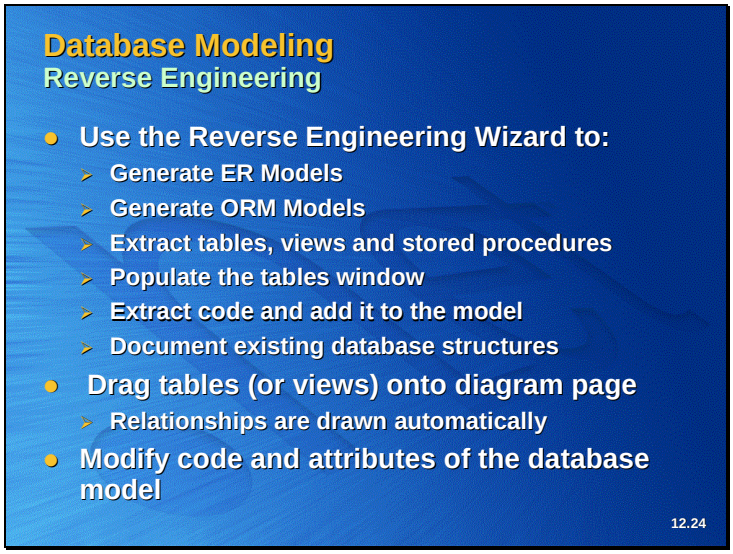
Entity Relationship diagrams are easy to work with in Visio. Some of the key windows in the ER Diagram interface include the Tables Window, Stencils pane, and the Database Properties Window.

Tables Window
The Tables Window provides access to tables and views that are present in the model.

Stencils Pane
The Stencils pane provides the capability to add more entities or relationships on the model.

Database Properties Window
The Database Properties window provides greatest degree of access to model information. We can specify items such as column names, data types, index structures, relationship definitions etc for the object that is currently selected in the model.
In addition, it provides ability to add defaults to the columns.

To create an ER diagram, choose a drawing type - Database Model, ER Source etc. and then draw the appropriate entities and relationships on the workspace by dragging and dropping from the stencils pane. Finally, change the names and attributes of the relationships and entities by double-clicking the item.


Database Modeling
Reverse Engineering
Use the Reverse Engineering Wizard to:
Generate ER Models
Generate ORM Models
Extract tables, views and stored procedures
Populate the tables window
Extract code and add it to the model
Document existing database structures
Drag tables (or views) onto diagram page
Relationships are drawn automatically
Modify code and attributes of the database model
12.24

Reverse Engineering

Reverse Engineering is a built-in feature of Visio that is integrated with Visual Studio .NET Enterprise. It allows you to easily generate ER or ORM models from existing database structures that were previously undocumented. This documentation can also be used for making decisions about the future direction that the database structure needs to take.

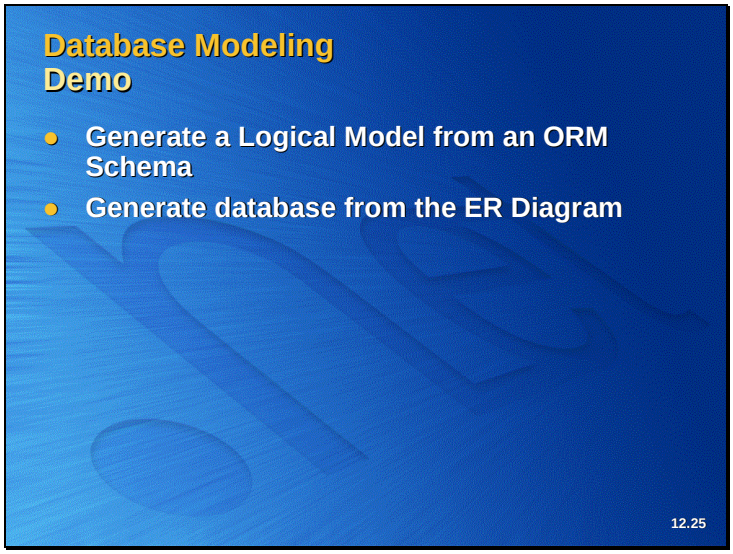
The Wizard allows you to capture physical database information such as primary keys, indexes, foreign keys, triggers, etc. In addition you can also extract database object information such as views, stored procedures, etc.

When you run the reverse engineering wizard, it populates the tables window with all of the tables and views that were extracted from the database. Tables and views can then be simply dragged out onto the model. In addition, you have the capability to simply right click an entity and choose **Show relationships** to follow the flow of database relationships that are present.

Also, the wizard generates the code for objects such as stored procedures, triggers, functions, check clauses, view definitions, and raw DDL.

It is easy to modify code that is extracted from the reverse engineering wizard as well as to modify attributes of tables, columns, relationships etc. that have been extracted.

 ©2002 Microsoft Corporation



Database Modeling

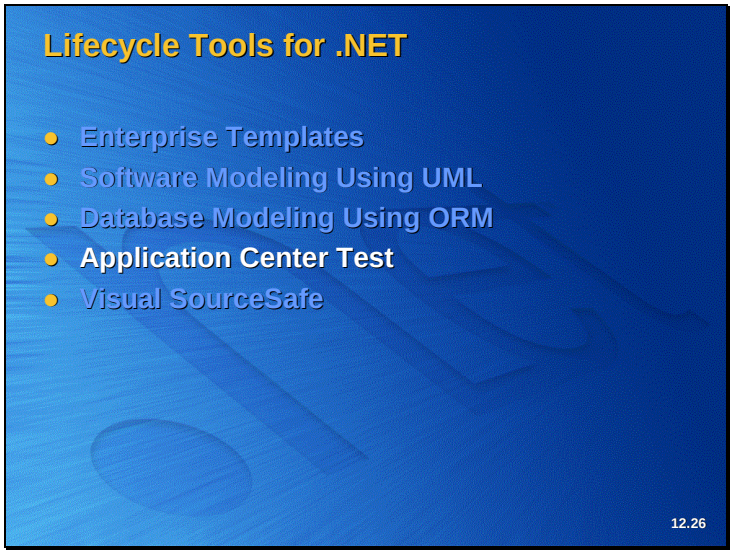
Demo

Generate a Logical Model from an ORM Schema

Objectives:

- To reverse engineer an ORM conceptual model from an existing database and then generate a Logical Model from the ORM model.
- Generate Physical database from the ER Diagram.

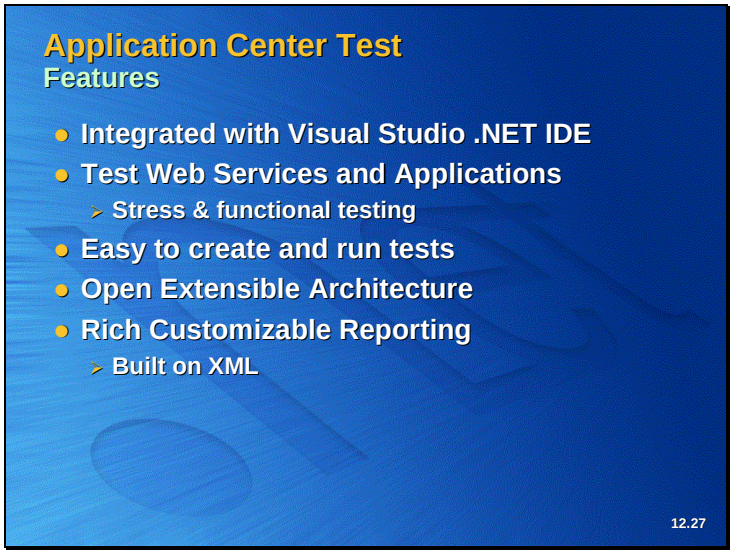
Note: The Northwind database will be used for this demonstration.



Lifecycle Tools for .NET

- Enterprise Templates
- Software Modeling using UML
- Data Modeling using ORM
- ▶ **Application Center Test**
- Visual SourceSafe

 ©2002 Microsoft Corporation



Application Center Test (ACT)

Application Center Test integrated with Visual Studio .NET enables developers to gather performance metrics and perform functional testing of XML Web services and applications within the Visual Studio .NET development environment.

Application Center Test includes features for performing stress, load and functional testing of XML Web services and applications. Application Center Test is integrated into the Visual Studio.NET IDE. This provides a cost effective way for organizations to incorporate testing early in a project and to perform testing as part of their overall application life cycle. Tests generated by developers can be easily reused by testing and operations groups and automated for maximum efficiency and coverage. All of this results in a more predictable and supportable application development process.

Test Web Services and Applications
Application Center Test is designed to Stress test Web servers and analyze performance and scalability problems with Web applications, including the ASP pages and the components they use. ACT simulates a large group of users by opening multiple connections to the server and rapidly sending HTTP requests.

Application Center Test provides long-duration and high-load stress testing capabilities. The programmable Dynamic Tests is useful for functional testing.

Easy to Create and Run Tests
Application Center Test provides two basic test types:

- **Static**
  Static tests use a grid interface similar to Microsoft Excel that does not require programming to create. These tests are produced using the main ACT user interface.
- **Dynamic**
  Dynamic tests, on the other hand, utilize an object model that is specifically designed for creating stress tests and that can be edited directly from the Visual Studio .NET IDE. These tests can be created using Microsoft Visual Basic Scripting Edition (VBScript), Microsoft JScript .NET development software or Perl.

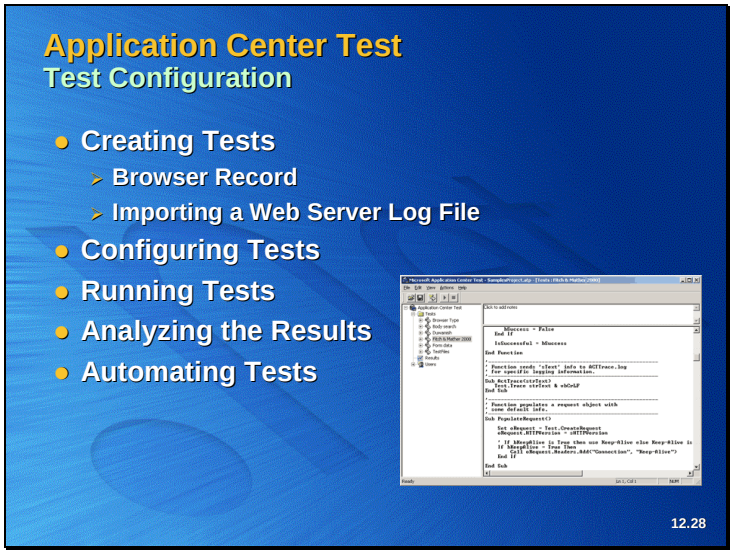
Excellent feature of Application Center test is its ability to convert a static test into a dynamic test. Dynamic tests provide greater flexibility because they allow the script to analyze the response from the Web server and take conditional action.

Open Extensible Architecture
Application Center Test provides sophisticated scripting capabilities, so that developers can analyze relative performance by simulating many simultaneous page requests to an XML Web service or application. This helps developers design faster, more stable XML Web services and applications.

Application Center Test supports several different authentication schemes and the SSL protocol, making it ideal for testing personalized and secure sites. Application Center Test is compatible with all Web servers and Web applications that adhere to the HTTP protocol. Since HTTP request headers and their values are customizable with most test types, applications using HTTP extensions, such as Distributed Authoring and Versioning (WebDAV), can also be tested.

Rich Customizable Reporting
Application Center Test provides a host of reporting features, charts, and comparisons, including the ability to save the test results in XML. You can run XSL transforms against your test data and extract the data. With XML, it is easy to access, extrapolate, and share the data.

 ©2002 Microsoft Corporation



Using Application Center Test

Creating Tests

Developers can create the test by using the following methods:

Browser Record Test

The Browser Record test-creation method is invoked from the main Application Center Test user interface. When developers create a browser record test, ACT captures all of the HTTP requests sent by their browser as they navigate a Web application. This makes it easy to create an accurate test because it captures all the inline images that would otherwise be automatically requested by the browser. Developers have the option of creating either a static or a dynamic browser record test.

If a dynamic test is chosen, the Visual Studio .NET IDE can be used to edit and run the test, making it easy to code the application while obtaining functional and performance information within the same environment.

Importing a Web Server Log File

If a Web server log is available, i.e., a log created from the live production site, it can be imported into the Application Center Test to create a test that accurately simulates real-world Web traffic. Web server log files are imported using the main Application Center Test user interface.

Web server log files import the following formats:

- Microsoft Internet Information Services (IIS) log file format
- NCSA Common log file format
- W3C Extended log file format

Configuring Tests

Developers can use the test properties to set the number of simultaneous requests that are generated and the uniqueness of each request. Each test type allows the developer to specify either the exact number of simultaneous browser connections or a target number of requests per second that the tool automatically achieves. Tests are run for either a number of iterations or a length of time.

The property pages for a test also control whether the tool automatically creates unique users or cycles through a predefined list of user names and passwords. This option is useful for Web sites that require specific user names to be authenticated.

Running Tests

Once the test is created and configured, developers must right-click the test name and select **Run Test** to execute a test run. While the test runs, the tool displays run-time data that provides a visual status of the request rate and feedback on any errors that may occur during the stress test.

Analyzing the Results

Understanding the results of performance tests is critical to the successful tuning of an XML Web service or application. Application Center Test provides a rich set of standard reports that are built using XML and that can be easily customized by developers to depict the information pertinent to their applications.
The default information that is available while running tests in the Visual Studio .NET IDE includes the following:

- Requests per second
- Time to first byte (response-time metric)
- Time to last byte (response-time metric)
- TCP and HTTP errors
- Number of satisfied requests

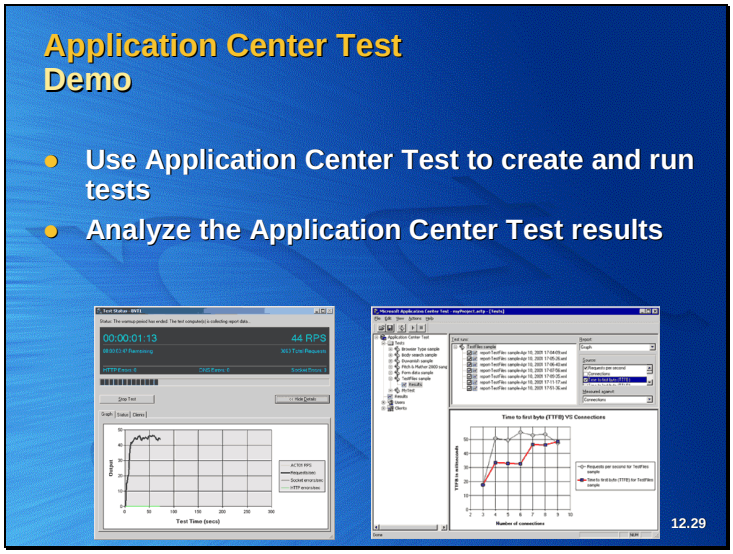
The main Application Center Test interface contains more detailed information regarding the results of a test. The Results section contains several predefined charts that display the most commonly requested performance metrics.

Developers also can use the custom chart option to create a chart that contains metrics from several different tests as well as any number of runs that have occurred from within all tests. In addition, the results contain tabular data that displays the relative performance of all the pages that were tested, as well as general test information and settings.

Automating Tests

Application Center Test enables developers to automate the test process by using the application object model that controls the ACT test tool itself. The application model is accessible from Microsoft Visual Basic for Applications (VBA), other scripting languages that understand COM, and any of the Visual Studio .NET languages. This allows developers to create powerful sets of tests that can be run automatically, and dramatically simplifies the process of running regression tests and other routine tasks.

 ©2002 Microsoft Corporation


Application Center Test
Demo
Use Application Center Test to create and run tests
Analyze the Application Center Test results
12.29

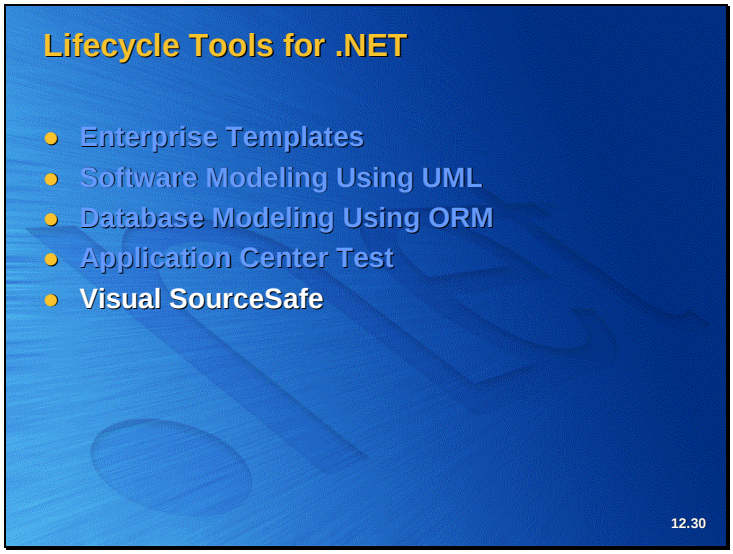
Application Center Test

Demo

Using Application Center Test, perform a Dynamic test against a Web Service and analyze its test summary.

Objectives:

- Use Application Center Test to create and run tests
- Analyze the Application Center Test results



Lifecycle Tools for .NET

- Enterprise Templates
- Software Modeling using UML
- Data Modeling using ORM
- Application Center Test
- ▶ **Visual SourceSafe**

 ©2002 Microsoft Corporation

Visual SourceSafe

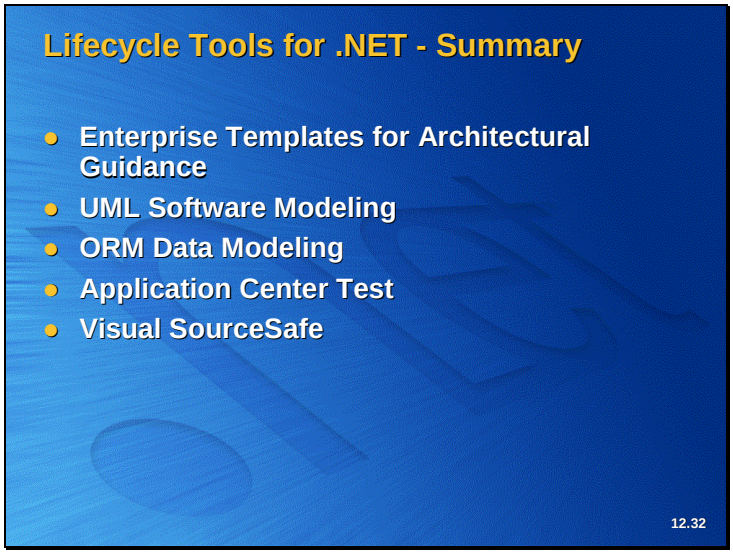
Visual SourceSafe (VSS) is a version-control system, it allows development teams to securely share application source code, design documents, and other project assets.

The key features of Visual SourceSafe include Version Control, Tracking Changes and Parallel Development. These features are discussed below.

Version Control
Version control is a critical facet of team software and Web site development, prevents users from accidental file loss, allows backtracking to previous versions, branch, merge, and manage releases. Visual SourceSafe provides this robust versioning and configuration management functionality along with security and audit trail capabilities.
Also, Visual SourceSafe is a source control system. Source control is highly desirable when developing software, Web sites, or anywhere you may want to compare two versions of a file or return to a previous version.

Tracking Changes
The Pending Check-ins Window feature tracks the changes a developer makes to files that are checked out, enabling those changes to be located even before the files are checked back in for others to use. With Visual SourceSafe, files can be checked out up to 10 times faster using RAS, and up to twice as fast over a LAN.

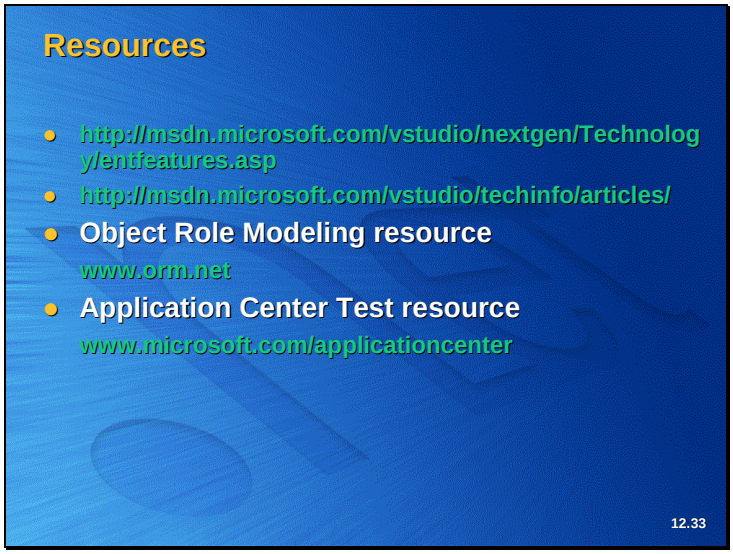
Parallel Development
Rich branching capabilities support parallel development and increase overall effectiveness, allowing organizations to work on multiple versions of an application at the same time.
Visual SourceSafe provides the branching and merging features necessary to support parallel development. Features like Drag and drop, file sharing, and branching make it easier to create parallel development environments.



Lifecycle Tools for .NET – Summary

- Enterprise Templates for Architectural Guidance
- UML Software Modeling
- ORM Data Modeling
- Application Center Test
- Visual SourceSafe

 ©2002 Microsoft Corporation



Resources

- http://msdn.microsoft.com/vstudio/nextgen/Technology/entfeatures.asp
- http://msdn.microsoft.com/vstudio/techinfo/articles/

 ©2002 Microsoft Corporation